Processamento da Informação

Tipos de Dados, Variáveis, Entrada e Saída

Prof. Rogério Neves

rogerio.neves@ufabc.edu.br

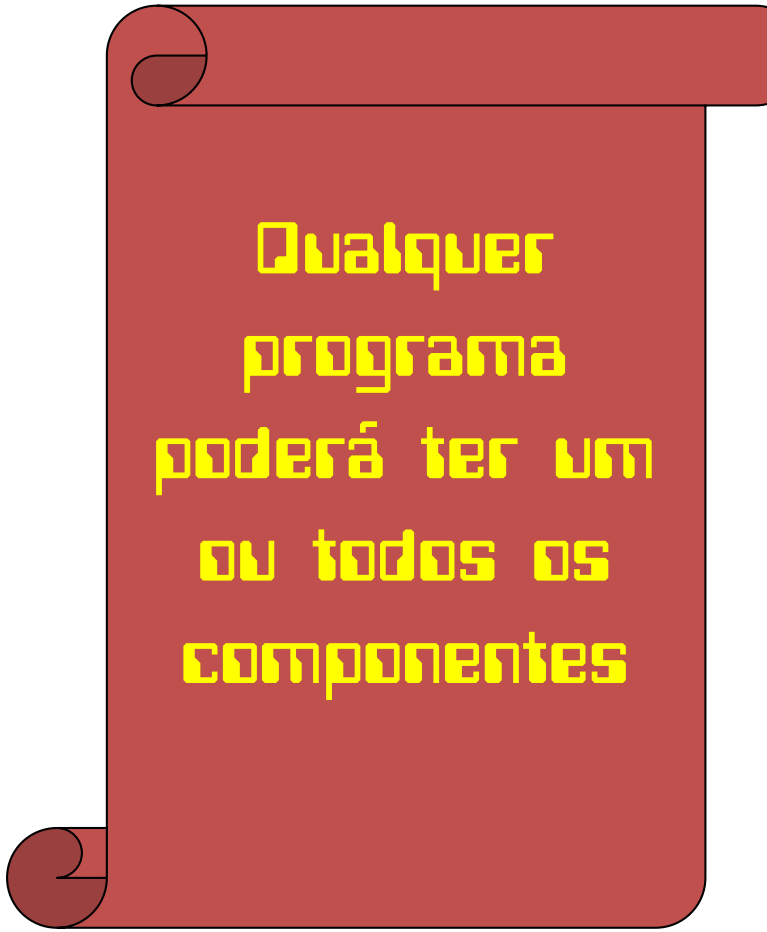
Roteiro da Aula

- Estrutura de um algoritmo
  - Introdução ao JAVA
- Tipo de dados
- Constantes
- Operadores lógicos e relacionais
- Comandos de saída
- Variáveis
- Comandos de entrada

ESTRUTURA DE UM ALGORITMO

Estrutura

- Constantes
- Variáveis
- Operadores de Entrada / Saída
- Operador de Atribuição
- Operadores Matemáticos e Lógicos
- Estruturas de Decisão
- Estruturas de Repetição



ANALOGIA....

- FACILITA PENSAR NO PSEUDOCÓDIGO
- FOQUE NA LÓGICA! DEPOIS PENSE NA SINTAXE

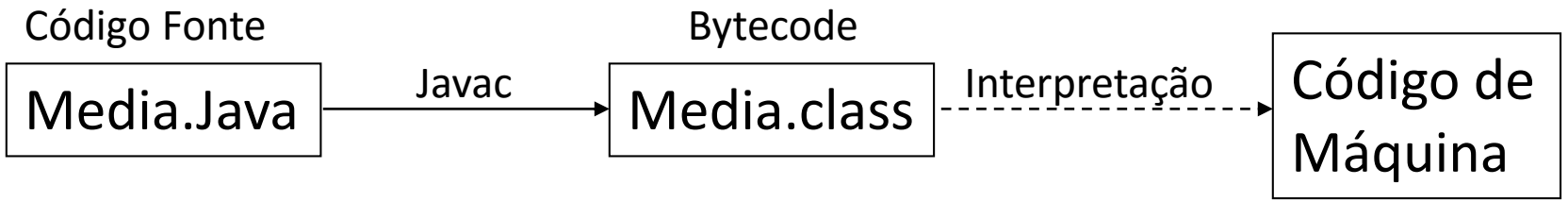
Este símbolo vai indicar a sintaxes java!

Linguagem Java

- Desenvolvida pela Sun Microsystem
  - ORACLE comprou a SUN
- **Portável** entre diferentes plataformas
- **Orientada a Objetos**
- **Multithread**: Permite que o programa execute em mais de um thread (linha de execução)
- **Interpretado**

Linguagem Java

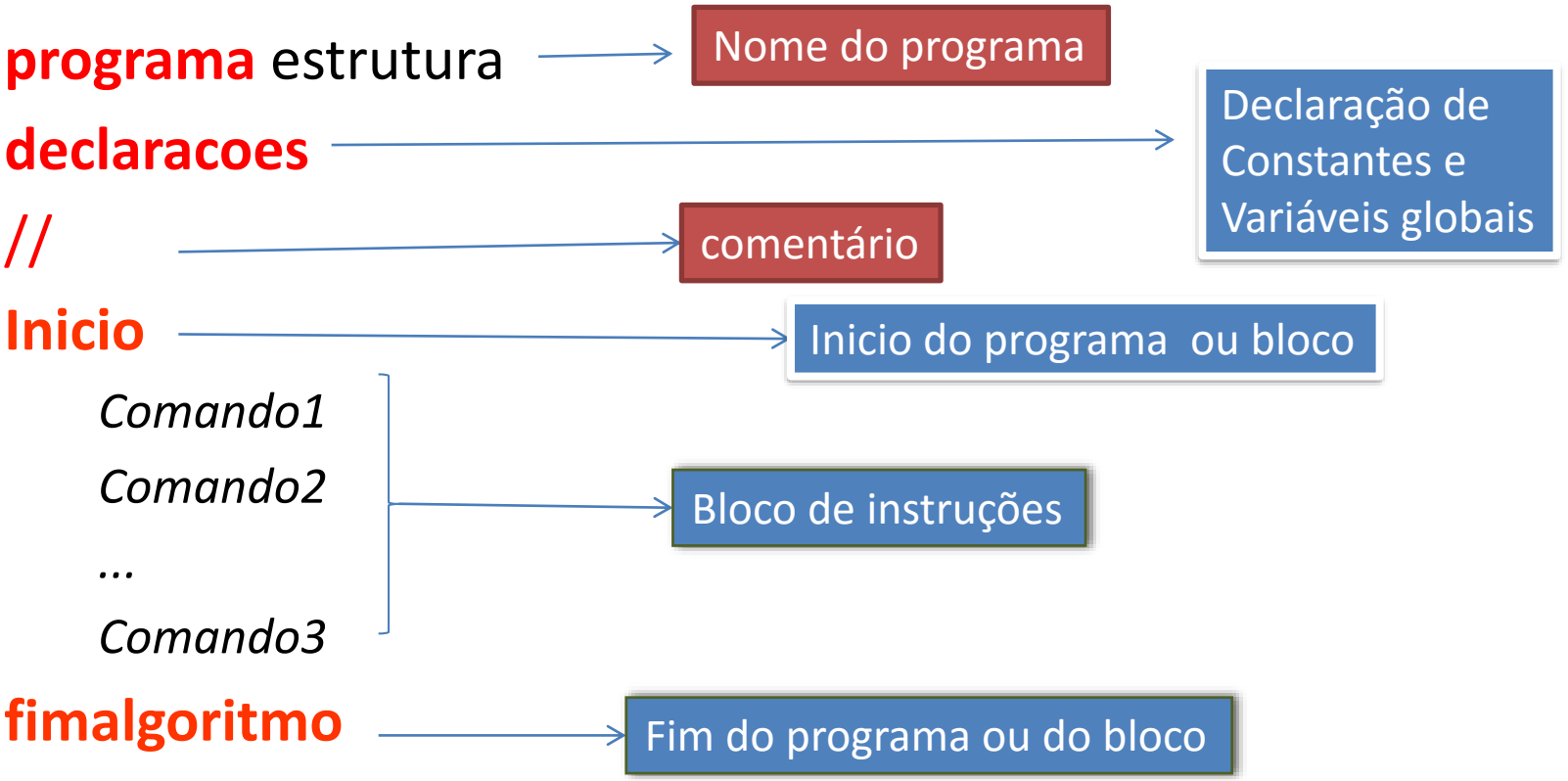
- Ambiente Java de Compilação:
  - Compilador JAVAC:



Os detalhes serão abordados em laboratório

Notação utilizada na aula
PSEUDOCÓDIGO

- Nesse curso, os algoritmos terão a seguinte estrutura



Estrutura adotada EM JAVA


```
Class estrutura {

    public static void main(String args[])
    {
 //aqui tem um comentário
        instrução 1;
        instrução 2;
        instrução 3;
            …
        instrução n;
    }
}
```

BOAS PRÁTICAS

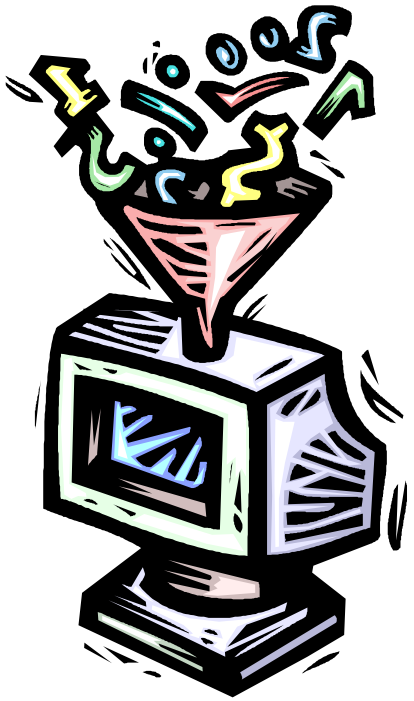
1- recue o corpo inteiro de cada definição de bloc um "nível" de recuo entre { e } que definem o corpo do método.
2- Comente seu código
3- Coloque nomes mnemônicos para as variáveis

Boas práticas também serão cobradas!

TIPO DE DADOS

Tipos básicos de dados

O que é possível guardar dentro
do computador???



TIPOS DE DADOS

PRINCIPAIS TIPOS DE DADOS EXISTENTES

Descrição	**Em PSEUDOCÓDIGO**	**Em Java**
Caractere	caractere	char
Literal	cadeia	String (classe)
Inteiro	inteiro	int
Inteiro longo	Não se aplica	long
Real (ponto flutuante)	real	float
Real (ponto flutuante)	real	double
Booleano	logico	boolean

Tipos básicos de dados

- ***Inteiro (int):*** números inteiros sem parte fracionária, podendo ser negativo, nulo ou positivo
  - a-) Ele tem 15 irmãos
  - b-) A escada possui 8 degraus
  - c-) Meu vizinho comprou 2 carros novos
- ***Real (float , double):*** números com parte fracionária, podendo ser negativo, nulo ou positivo
  - a-) Ela tem 1.73 metro de altura
  - b-) João pesa 85.5
    - float – 7 casas de precisão após a vírgula
    - ***double*** – 15 casas de precisão após a vírgula

Tipos básicos de dados

- ***Literal (char , String):*** conjunto de caracteres alfanuméricos números (0.. .9), letras (A.. .Z, a.. .z) e símbolos (#, ?, !, @.....)
  - a) Constava na prova: “Use somente caneta!”
  - b) O parque municipal estava repleto de placas: “Não pise na grama”
  - c) O meu e-mail é: “fulano@ufabc.edu.br”
  - d) ‘A’
    - **Para um caracter usar aspas simples**
    - **Para um conjunto de caracteres usar aspas dupla**

Tipos básicos de dados

- ***Lógico (boolean):*** poderá assumir valores Verdadeiro ou falso
  - ***true*** – verdadeiro
  - ***false*** – falso

Exercícios

- Indique o tipo dos dados abaixo:
  1. 1000
  2. “-9000”
  3. “true”
  4. 678
  5. 45.8976
  6. -1502
  7. false

CONSTANTES

Constante

- **Definição**
  - Valor fixo que **NÃO** se modifica durante a execução de um programa
- **Exemplos**
  - Número
  - Valor Lógico (Verdadeiro ou falso)
  - Seqüência de Caracteres
- **Classificação**
  - Numérica
  - Literal
  - Lógica

Constante Numérica

- Nos algoritmos utiliza a notação decimal
- As constantes numéricas podem
  - Possuir ou não uma parte fracionária
  - Ter uma parte exponencial
    - Fator 10 elevado a um número inteiro
- Exemplos
  - 25
  - 3.14
  - 7.8 E10 (idêntico a $7.8 * 10^{10}$)

Constante Numérica

- Pode ser positiva ou negativa
  - Depende do sinal que precede a constante
    - Caso não exista sinal → positiva
  - Expoente também pode possuir um sinal (indica o deslocamento da virgula)
    - Caso não exista sinal → positiva
- Exemplos
  - -3.4
  - 26E-10 (idêntico a $26*10^{-10}$)
  - **3,4 (Erro não representa um número em java)**

Os números fracionários sempre serão representados através da notação de ponto

Constante Lógica

- Só pode ser
  - Verdadeiro TRUE 1
  - Falso FALSE 0
- Utilizado em preposições Lógicas

Constante Literal

- Qualquer seqüência de caracteres
  - Letras, dígitos, símbolos Especiais
- Em java Todas as constantes literais que aparecem no algoritmo devem estar entre **aspas**
- Exemplos:
  - “José da Silva”
  - “Mensagem”
  - “12345”
  - “@#!ABC4”
  - “26/02/75”
  - “X1W!Z2”

Constantes Literais

- **Não confunda!!!!**
  - 12345
  - “12345”
  - false
  - “false”

Exercícios

- Identifique cada um dos tipos das constantes abaixo:

a) 21

b) “Bola”

c) “true”

d) 0.21 E2

e) false

Operadores

- São meios pelos quais
  - incrementamos,
  - decrementamos,
  - comparamos e
  - avaliamos dados dentro do computador
- Temos três tipos de operadores:
  - Operadores Aritméticos
  - Operadores Relacionais
  - Operadores Lógicos

Operadores Aritméticos

Descrição	**Em Pseudocódigo**	**Em Java**
Multiplicação	*	*
Divisão real	/	/
Divisão inteira	div	/
Módulo	mod	%
Adição	+	+
Subtração	-	-
Incremento	Não se aplica	++
Decremento	Não se aplica	--

Exemplo

- Qual o resultado da avaliação da expressão

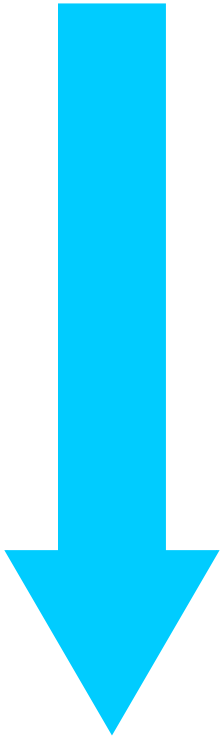
10 - 4 * 2 + 1

- 13, 18, 3 ?

- Depende da prioridade da avaliação dos operadores

Ordem de Prioridade

Menor



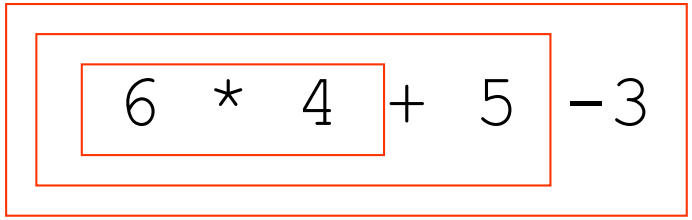
+ -

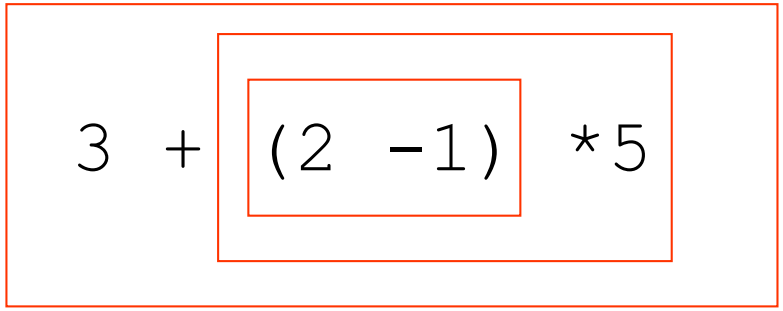
/ *

( )

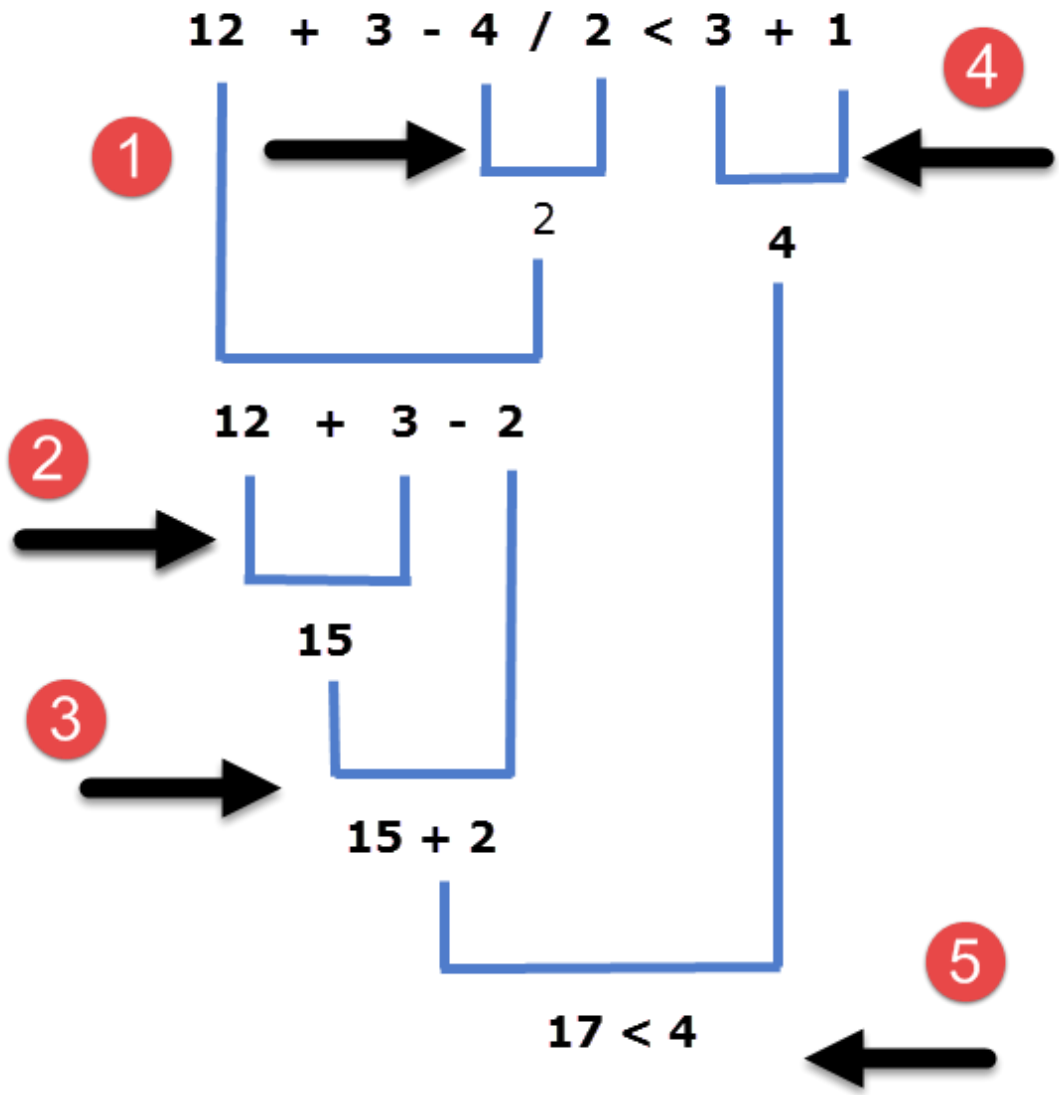
Maior

Exemplos





Precedência de Operadores



false

CONVERTENDO TIPOS NUMÉRICOS

(var) Type Cast

```
int inteiro = 1;
float f1 = 1.1f, f2, f3;
// float precisa de f para diferenciar de double

double d1 = 2.2, d2, d3;


f2 = inteiro;
f3 = (float)d1;


d2 = inteiro;
d3 = f1;
inteiro = (int)d1;
```

Conversões

- Inteiro para real: automático
- Qualquer real para inteiro: requer type cast

```
inteiro = (int)real;
```

- float para double: automático
- double para float: requer type cast

```
f = (float)d;
```

Regra geral

Se houver perda de precisão → usar type cast

OPERADORES LÓGICO /RELACIONAIS

Operadores Lógicos

- Estabelecem uma relação de comparação entre valores ou expressões
- Resultam sempre em um valor lógico
  - **verdadeiro** ou **falso**

Operadores Lógicos

Descrição	**Em Pseudocódigo**	**Em Java**
E	e	&&
OU	ou	||
NÃO	não	!

Operadores Lógicos

- Os operadores lógicos mais utilizados são:
  - E → **&&**
  - OU → **||**
  - NÃO → **!**
  - Tais operadores retornam valores lógicos como **(V)erdadeiro** ou **(F)also**

‘Se tiver macarronada ou frango → Eu vou almoçar ’
Quando eu vou almoçar?

‘Se chover e relampejar, eu choro de raiva!’
Quando eu choro de raiva?

Operadores Lógicos

- Tabela-verdade para os operadores **&&** e **||**

A	B	A && B	A || B
V	V	**V**	V
V	F	F	V
F	V	F	V
F	F	F	**F**

&& – somente resulta em VERDADEIRO quando <u>todas</u> as sentenças avaliadas são <u>verdadeiras</u>

|| – somente resulta em FALSO quando <u>todas</u> as sentenças avaliadas são <u>falsas</u>

Operadores Lógicos

- O operador ! (não) faz a negação de uma

‘Pedro mora na Vila Rica’

- ‘Pedro NÃO mora na Vila Rica’ ssa a ser:

A	! A
V	F
F	V

Não é verdade que não é verdade que não é verdade que não é verdade que ‘Maria é casada?’

VERDADE ou MENTIRA?

Precedência dos operadores: !, && ou ||

Operadores Relacionais

Descrição	**Em Pseudocódigo**	**Em Java**
Maior	>	>
Maior ou igual	>=	>=
Menor	<	<
Menor ou igual	<=	<=
Igualdade	=	= =
Desigualdade	!=	! =

Linguagem Java

- **Operador de Incremento em Java:**
  - **Função:** Incrementar de 1 o operando
  - Trabalha de dois modos:
  - Pré-fixado ➔ ++num
    - A variável num é incrementada antes de seu valor ser usado
  - Pós-fixado ➔ num++
    - A variável num é incrementada depois de seu valor ser usado

Pré-fixado

```
num = 5;
x = ++num;
```

x=6 num=6

Pós-fixado

```
num = 5;
x = num++;
```

x=5 num=6

Linguagem Java

- **Operador de Decremento em Java:**
  - **Função:** Decrementar de 1 o operando
  - Trabalha de dois modos:
  - Pré-fixado ➔ --num
    - A variável num é decrementada antes de seu valor ser usado
  - Pós-fixado ➔ num--
    - A variável num é decrementada depois de seu valor ser usado

Pré-fixado

```
num = 5;
x = --num;
```

x=4 num=4

Pós-fixado

```
num = 5;
x = num--;
```

x=5 num=4

Precedência de Operadores JAVA

Operador	Observação
( ), [ ]	Parêntese e Colchetes para agrupar expressões.
*, /	Operadores Aritméticos de multiplicação e divisão.
+, -	Operadores Aritméticos de adição e subtração.
= =, <, >, <=, >=, !=	Operadores relacionais.
!	Operador lógico de negação.
&&	Operador lógico e.
||	Operador lógico ou.
=	Operador de atribuição.

Exercícios

1 - Determine o resultado lógico das expressões, considerando os seguintes valores: X=1, A=3, B=5, C=8 e D=7.

a) ! (X > 3)

b) (X < 1) && ! (B > D)

c) ! (D < 0) && (C > 5)

d) ! (X > 3) || (C < 7)

e) (A > B) || (C > B)

f) ! (D > 3) || ! (B < 7)

Exercícios

2 - Considerando X = 4 e Y = 5, avalie as expressões abaixo e classifique o resultado como verdadeiro ou falso:

a) X == 4 **&&** Y == 7

b) X < 3 **||** Y != 7

c) X >= 2 **&&** Y ==5

d) **!** (X != 2) **&&** Y > 4

e) X < 5 **&&** Y > 2 **||** X != 7

Linguagem Java

- **Operadores:**
  - **Exercícios:**

1. Faça o teste de mesa para encontra o valor final das variáveis a, b, c e x. Depois escreva um programa em C para analisar as instruções abaixo:

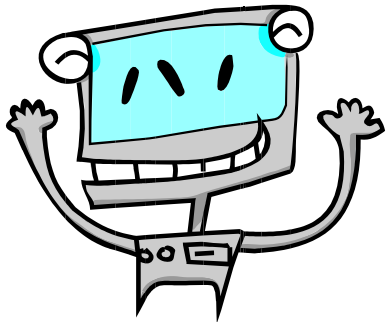
```
a=3;
b=4;
c=5;
x=a++ * b;
b=a;
x=100 / (--b * c);
c++;
```

a	b	c	x

COMANDOS DE SAÍDA

Comando de Saída

- Todos os algoritmos devem executar um determinado processamento
- O resultado do processamento deve ser apresentado para o usuário através de um dispositivo de saída
- Geralmente os dados são apresentados na tela do computador
  - Dispositivo de saída padrão



COMANDO DE SAÍDA PSEUDOCÓDIGO

```
programa Alo
inicio
    escreva (“alo mundo“)
fim
```

Lembre-se! String vem entre aspas!

Comando de Saída

- **Utilizado para escrever um resultado de um processamento na saída padrão (monitor)**

Pode escrever constantes, variáveis e resultado de um cálculo

Exemplo

```
class Alo{
   public static void main(String args[ ]){
     System.out.println("Alo Mundo!!!!!");
   }
}
```

System.out

Dispositivo de saída padrão (monitor, console,...)

Qual a saída?

```
System.out.print(“1”);
System.out.print(“2”);
System.out.println(“3”);
System.out.println(“4\n”);
System.out.println(“5”);
```

Qual a saída?

```
System.out.print(“1”);
System.out.print(“2”);
System.out.println(“3”);
System.out.println(“4\n”);
System.out.println(“5”);
```

```
123
4

5
```

Print formated (printf)

```
String nome = "João";
int idade =15;
float altura =1.55f;
System.out.printf(
   "Meu nome é %s.\n"
   + "Eu tenho %d anos "
   + "e %.2fm de altura\n",
   nome, idade, altura);
```

Print formated (printf)

```
String nome = "João";
int idade =15;
float altura =1.55f;
System.out.printf(
   "Meu nome é %s.\n"
   + "Eu tenho %d anos "
   + "e %.2fm de altura\n",
   nome, idade, altura);
```

QUAL A SAÍDA?

Print formated (printf)

```
String nome = "João";
int idade =15;
float altura =1.55f;
System.out.printf(
   "Meu nome é %s.\n"
   + "Eu tenho %d anos "
   + "e %.2fm de altura\n",
   nome, idade, altura);
```

```
Meu nome é João.
Eu tenho 15 anos e 1.55m de altura
```

```
System.out.printf("Variaveis byte: %d e %d\n", byte1, byte2);
/*                 "String          ↑    ↑  ", parâmetros
                                   Máscara dos parâmetros
... Siga o exemplo para os demais tipos de acordo com a tabela:
Usando a mascara para formatar dados, sempre começando com %:
%d – decimal inteiro          %u – inteiro positivo
%f – real formato xxx.yyyyyy  %E – real científico x.yyyyyyE+zzz
%c – caractere                %X – %d em hexadecimal maiúsculo
%s – String                   %x – %d em hexadecimal minúsculo
%b – booleano                 %o – %d em octal

Modificadores:
     %[-][+][0-N][.0-9][lL][dxXuofeEgGcs]
     ||  |  |    |     |   |
     ||  |  |    |     |   +- formato (veja tabela acima)
     ||  |  |    |     +----- modificador long, ignorado
     ||  |  |    +----------- número de casas decimais após o .
     ||  |  +---------------- largura do campo (N caracteres)
     ||  +------------------- mostra '+' para números positivos
     |+---------------------- alinha a esq.
     +----------------------- inicia formatação
```

printf

Usando a mascara para formatar dados, sempre começando com %:

%d – decimal inteiro	%u – inteiro positivo
%f – real formato xxx.yyyyyy	%E – real científico x.yyyyyyE+zzz
%c – caractere	%X – %d em hexadecimal maiúsculo
%s – String	%x – %d em hexadecimal minúsculo
%b – booleano	%o – %d em octal

Modificadores:

```
%[-][+][0-N][.0-9][lL][dxXuofeEgGcs]
||  |  |    |     |   |
||  |  |    |     |   +- formato (veja tabela acima)
||  |  |    |     +----- modificador long, ignorado
||  |  |    +----------- número de casas decimais após o .
||  |  +---------------- largura do campo (N caracteres)
||  +------------------- mostra '+' para números positivos
|+---------------------- alinha a esq.
+----------------------- inicia formatação
```

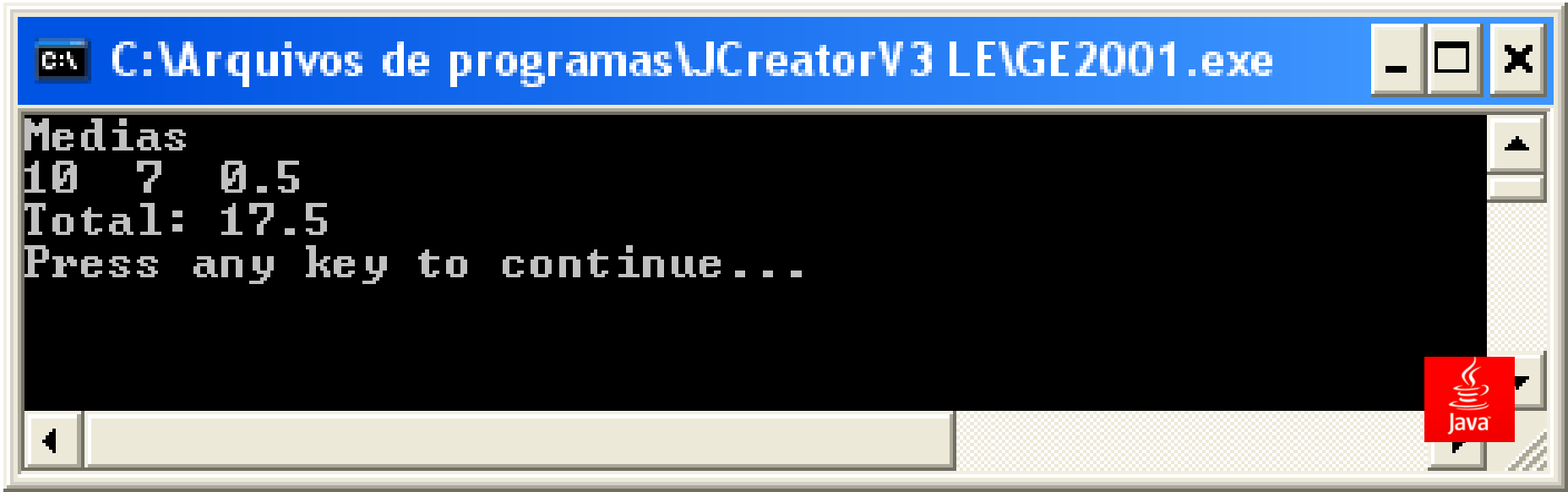
printf("d= %+5.2f", 2.0)

Comando System.out.println( )

- Para escrever várias informações devemos separá-las pelo operador de **concatenação** (**+**)

Exemplo

```
System.out.println("Medias");
System.out.println(10 + "  " + 7 + "  " + 0.5);
System.out.println("Total: " + 10 +  7 +  0.5);
```



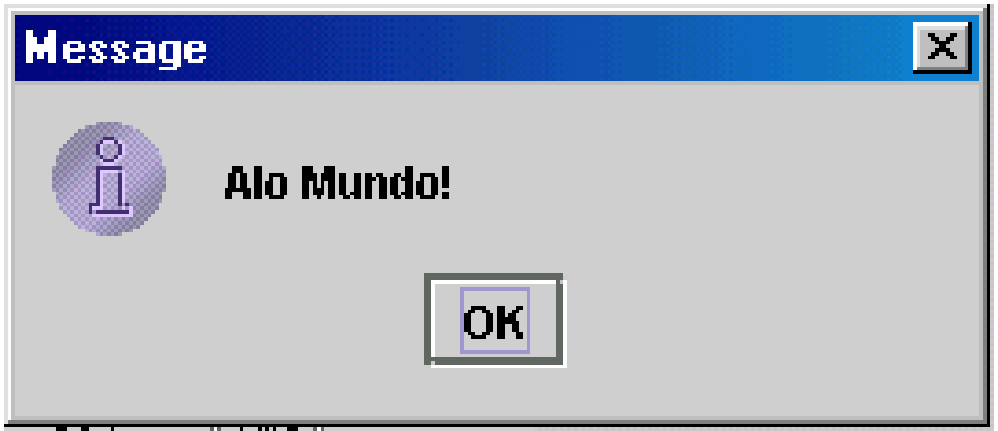
Saída com Interface Windows

```
import javax.swing.*;
class PrimeiroB {
  public static void main ( String args [ ]
  ){

  JOptionPane.showMessageDialog(null,"Alô
  Mundo!");
  }
}
```

Tem que importar Biblioteca! Atenção!

Saída:



Exercícios

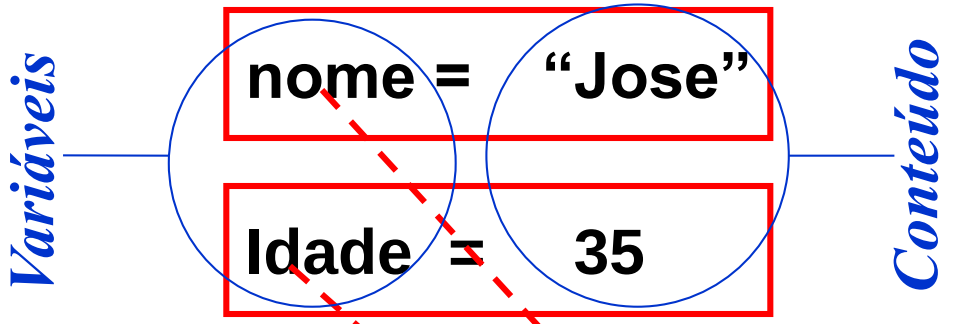
- 1 - Construir um algoritmo para calcular a média das seguintes notas 7.5, 4.5 e 9.
- 2 - Construir um algoritmo para calcular a área de um quadrado de 350 m de lado.
- 3 - Construir um algoritmo para calcular a área de uma circunferência com raio 5 cm. (ac=π*raio^2) π=3,14159.
- 4 - Construir um algoritmo para imprimir a **soma** de das seguintes strings “Hugo”, “gastou 50 reais”, “ontem”.

VARIÁVEIS

Variáveis

- **Variável**
  - É tudo aquilo que está sujeito a variações, que é inconstante
  - Serve para armazenar dados do programa na memória principal
  - Cada variável corresponde a uma **posição de memória**, cujo conteúdo pode se alterado ao durante a execução de um programa
  - Pode assumir apenas um valor **a cada** instante

Variáveis



Memória Principal

0000	.....
.....	.....
0100	Jose
0101	35
.....	.....
127Mb	.....

- Uma variável é um espaço reservado na memória para armazenar um tipo de dado

Regras para nomear variáveis em JAVA

- Pode conter um ou mais caracteres;
- Deve começar sempre por uma letra;
- Pode ser seguidos por letras e números;
  - casa21
- Não pode ter espaços entre as letras;
  - casa 21 (errado!!) – usar_underline_para_separar
- Não pode conter caracteres especiais;
  - (?, ç, @, #, !, etc....)
- Não pode ser uma palavra reservada;
  - float, int, for , ...
- Ser sucinto e utilizar nomes coerentes

Regras para nomear variáveis

- **Fique atento!!!**
  - Os nomes de variáveis abaixo não são iguais???


```
NomeCliente   nomecliente   nomeCliente
```

- **Não**, pois variáveis em Java são *case-sensitive*
  - nomes com letras maiúsculas são diferenciados de nomes com letras minúsculas!!

Exercícios

- Verifique se as variáveis abaixo possuem nomes corretos e justifique as alternativas falsas :

a) endereço

b) 21abril

c) fone$com

d) nomeusuário

e) nome_usuário

f) nome usuário

g) end*a-6

h) cidade3

i) #cabec

Exercícios

- Verifique se as variáveis abaixo possuem nomes corretos e justifique as alternativas falsas :

a) endereço

b) 21abril

c) fone$com

d) nomeusuário

e) nome_usuário

f) nome usuário

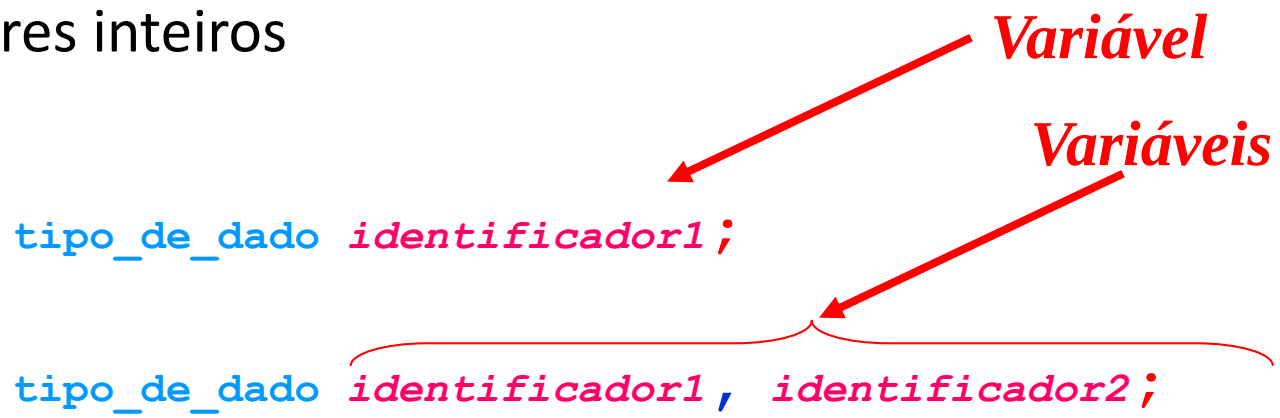
g) end*a-6

h) cidade3

i) #cabec

Declaração de Variáveis

- Para que as variáveis possam guardar algum valor, elas precisam ser **declaradas**
- Toda variável deve corresponder a um tipo base de dado, sendo assim uma variável do tipo inteiro só poderá armazenar valores inteiros



Em pseudocódigo

```
tipo_de_dado identificador1
tipo_de_dado identificador1, identificador2
```

Exemplos em Java

```
String nome , endereco;

float salario;

int  i , num;

double tamanho;

char opcao;
```


Atribuição

- É o mesmo que fornecer um valor a uma variável
- Quando uma variável tem um valor atribuído, ela irá guardar este valor até que seja modificada
- O tipo de dado **DEVE** ser compatível com o tipo da variável
- Portanto, se uma variável é do tipo String, somente podemos atribuir um valor do tipo String a ela
- Operador de atribuição que usaremos será (=)

```
Identificador1  =  valor;
```

Atribuição

- **Considere as variáveis abaixo**:

  int A;

  double X, B;

  double valor;

  String nome;

- **Exemplo de atribuição de valores:**
  - A = 85;
  - X = 8 + (13 / 5);
  - B = 100.52;
  - valor = 123.456789;
  - nome = “Joao da Silva”;

Exercícios

- 1 - Informe o tipo de dado mais apropriado para armazenar os seguintes dados, em seguida faça a declaração e a atribuição de valores:

a-) idade

b-) valor da conta telefônica

c-) nome de empresa

d-) media final na disciplina

Exercícios

3 - Defina os valores finais de A, B, C, D e X:


```
X = 0;
A = 10;
B = 20;
C = 30;
D = 40;
A = D + A;
A = D + B;
C = A;
A = D;
B = (B + B) + (B*A);
B = 40;
A = B – 10;
A = B + 1;
X = A + B + C + D;
```

Exercícios

- **3 - Defina os valores finais de A, B, C, D e X:**

```
X = 0;
A = 10;
B = 20;
C = 30;
D = 40;
A = D + A;
A = D + B;
C = A;
A = D;
B = (B + B) + (B*A);
B = 40;
A = B – 10;
A = B + 1;
X = A + B + C + D;
```

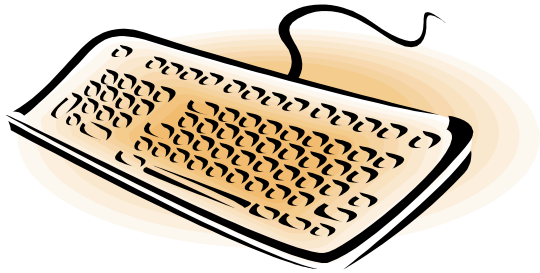
Teste de Mesa

X	A	B	C	D
~~0~~	~~10~~	~~20~~	~~30~~	40
	~~50~~			
	~~60~~			
			60	
	~~40~~			
		~~8~~40		
		40		
	~~30~~			
	41			
181				

COMANDOS DE ENTRADA

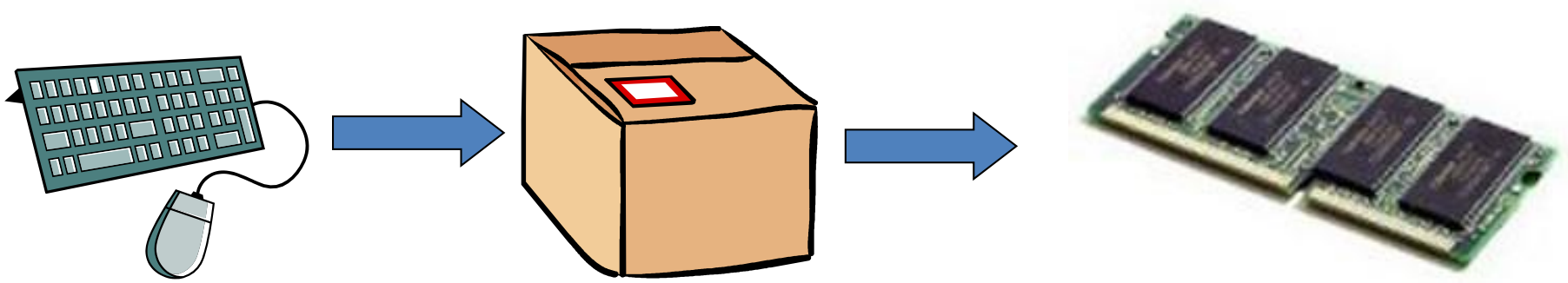
Comando de entrada

- Para que dados externos sejam processados é necessário ter uma forma de inserí-los no computador
- **Exemplo**:
  - Para calcular a média de duas notas quaisquer é necessário informar quais são as notas para que ocorra o processamento
- O dispositivo de entrada padrão é o **teclado** do computador



Comando de entrada

- **Mas**, como ler os dados do teclado e guardar na memória do computador?
  - É necessário criarmos um **buffer** para guardar os dados digitados no teclado e depois transferí-los para a memória (variável)



É o **buffer** do teclado

```
Scanner entrada = new Scanner(System.in);
```

Comando de entrada

- Após ter criado o **buffer** do teclado, basta transferir esse dado para a variável desejada
- A leitura é dada associando o tipo de entrada ao tipo da variável que receberá o dado

Tipo de Dado	Usar
String	buffer.**nextLine()**;
int	buffer.**nextInt()**;
double	buffer.**nextDouble()**;
float	buffer.**nextFloat()**;
char	buffer.**next()**.**charAt(0)**;
boolean	buffer.**nextBoolean()**;

Exemplo primeiro.java

```
import java.util.*;
class primeiro{
   public static void main(String args[]){
        /*declaração das variáveis*/
        Scanner entrada = new Scanner(System.in);
        int num;
        double salario = 0;

        /*corpo do programa*/
        System.out.println("Qual o aumento(%)?");
        num = entrada.nextInt();
        System.out.println("Qual o salario?");
        salario = entrada.nextDouble();
        salario = (salario * num)/100 + salario;
        System.out.println("o novo salário é: " + salario);
              }
}
```

Necessário importar as bibliotecas!

Comando de entrada

- **Problema:**
  - Alguns erros podem acontecer na leitura de tipos de dados numéricos e caracteres (String/char)
- **Solução:**
  - Trabalhar com 2 buffers
  - Criar um novo buffer para ler String / char
  - Usar o método `var = leia("");`

Método leia

```
public static float leia(String texto) {
        java.util.Scanner s = new java.util.Scanner(System.in);
        System.out.print(texto);
        return s.nextFloat();
}
```

Método leia

```
public static float leia(String texto) {
        java.util.Scanner s = new java.util.Scanner(System.in);
        System.out.print(texto);
        return s.nextFloat();
}
 public static void main(String[] args) {
       int n = (int)leia("Entre um número inteiro: ");
       double real = leia("Entre um real");
}
```

Método leia para Strings

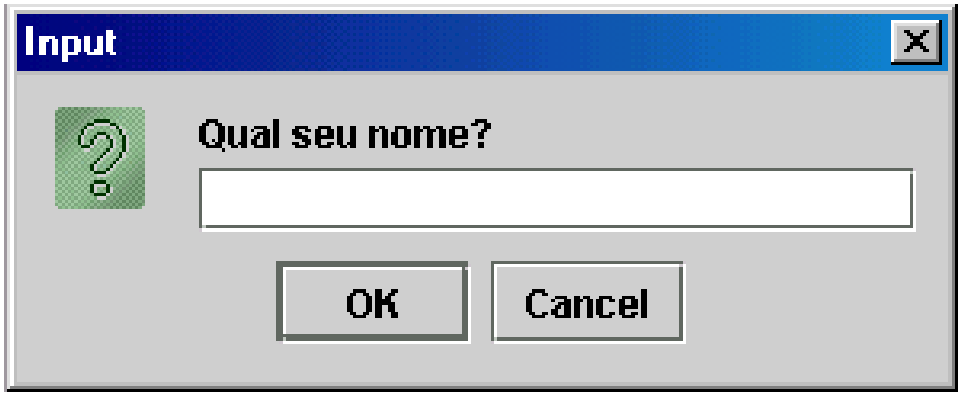
```
public static float leia(String texto) {
        java.util.Scanner s = new java.util.Scanner(System.in);
        System.out.print(texto);
        return s.nextFloat();
}
public static String leia(String texto, String tipo) {
        java.util.Scanner s = new java.util.Scanner(System.in);
        System.out.print(texto);
        return s.nextLine();
}
 public static void main(String[] args) {
       int n = (int)leia("Entre um número inteiro: ");
       double real = leia("Entre um real");
       String nome = leia("Qual é o seu nome? ", "");
}
```

Métodos leia para número e String

```
public static void main(String[] args) {
        int i = (int)leia("Entre com um valor inteiro: ");
        escreva("Valor entrado = " + i);
        float f = leia("real: ");
        double d = leia("real: ");
        System.out.printf("Valores entrados: %f e %f\n\n", f, d);
        String e = leia("Entre com uma String: ", "");
        escreva("String entrada = " + e);
    }
    public static void escreva(Object texto) {
        System.out.println(texto.toString());
    }
    public static float leia(String texto) {
        java.util.Scanner s = new java.util.Scanner(System.in);
        System.out.print(texto);
        return s.nextFloat();
    }
    public static String leia(String texto, String tipo) {
        java.util.Scanner s = new java.util.Scanner(System.in);
        System.out.print(texto);
        return s.nextLine();
    }
```

Comando de Entrada com Interface Gráfica

```
import javax.swing.*;
class EntradaB {
   public static void main ( String args [ ] ){
       String nome;
       nome  =  JOptionPane.showInputDialog(null,"Qual
   seu nome?");
   }
}
```



Comandos de Entrada

- **Comando de Entrada:**
  - Leitura retorna uma literal (String);
  - É necessário a conversão para o tipo indicado;
  - **Comandos para conversão de tipos:**

- ```
  int n =
  Integer.parseInt(JOptionPane.showInputDialog(
  "Digite um numero inteiro:"));
  ```

- ```
  float valor =
  Float.parseFloat(JOptionPane.showInputDialog(
  "Digite um numero real:"));
  ```

Comandos de Entrada

```
import javax.swing.*;
public class Le_Escreve{
  public static void main(String[] args) {
    float valor; // Declaracao de variaveis.
    int cont;
    String nome;
// Leitura de dados dentro de caixa de dialogo.
    nome = JOptionPane.showInputDialog("Nome do Aluno:");
    valor = Float.parseFloat(JOptionPane.showInputDialog (
                                    "Digite numero real:"));
    cont = Integer.parseInt(JOptionPane.showInputDialog (
                                    "Digite numero inteiro:"));
// Impressao de valores do tipo String, Real e Inteiro na tela.
    JOptionPane.showMessageDialog(null,"Nome digitado = "+nome);
    JOptionPane.showMessageDialog(null,"Real digitado = "+valor);
    JOptionPane.showMessageDialog(null,"Inteiro digitado = "+cont);
  }
}
```

Exercícios

1. Construir um algoritmo que calcule a área de um triângulo. (at= (altura*base) / 2)
2. Construir um algoritmo que leia quatro notas (real) e imprima a média aritmética
3. Construir um algoritmo que leia o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:
   - A idade dessa pessoa;
   - Quantos anos essa pessoa terá em 2050.

Exercícios

4. Sabe-se que um quilowatt de energia custa 1/500 avos do salário mínimo. Faça um algoritmo que receba o valor do salário mínimo e quantidade de quilowatts consumida por uma residência. Calcule e mostre:
   A. O valor, em reais, de cada quilowatt;
   B. O valor, em reais, a ser pago por essa residência
   C. O valor, em reais, a ser pago com desconto de 15%.